文章编号:1671-4598(2016)05-0274-05 DOI:10.16526/j.cnki.11—4762/tp.2016.05.076 中图分类号:TP393 文献标识码:A

基于活性顺序图的形式化验证方法及工具研究

张 坤,叶俊民,王 嫱,赵丽娟,陈 曙
(华中师范大学 计算机学院,武汉 430079)

摘要:近年来,形式化验证方法在软件开发过程的作用越来越大;如何充分利用形式化验证方法提高软件系统的可靠性已成为软件开发者及使用者主要关注的问题;总结了近年来基于活性顺序图的形式化验证方法的研究进展,首先介绍活性顺序图的语言及其表达能力与复杂性,然后深入分析现有的基于活性顺序图的形式化验证的关键技术及其典型应用,最后实现一种基于活性顺序图的运行时验证工具,实验证明使用本验证工具进行形式化验证的可行性。

关键词:活性顺序图;形式化验证;软件开发过程

Research on Formal Verification Method and Verification Tool Based on Live Sequence Chart

Zhang Kun, Ye Junmin, Wang Qiang, Zhao Lixian, Chen Shu
(School of Computer Science, Central China Normal University, Wuhan 430079, China)

Abstract: In recent years, the role of formal verification technology in the software development process is growing more and more important. How to use formal verification technology to improve the reliability of software systems is a major concerned problem of software developers and users. This paper summarizes the progress of formal verification method based on Live Sequence Chart recently. In this paper, the language of live sequence chart, its expressive power and complexity are first introduced. Then the existing key technologies and their application of formal verification method based on live sequence chart are analyzed. Finally, the runtime verification tool based on live sequence chart is implemented, experiments show the feasibility of using this verification tool to formal verification.

Keywords: live sequence chart; formal verification; software development process

0 引言

随着计算机技术的发展,软件系统已经渗透到人们的生活之中,软件系统关系着人民的信息安全、财产安全乃至生命安全。形式化验证方法在软件开发过程中的作用越来越受到软件开发者的重视,确保软件系统在任何时候都与需求规约一致,已经成为软件开发者及使用者关注的重要问题,利用场景对系统进行建模和分析已成为软件开发过程中形式化验证的重要技术和方法[1]。目前形式化验证方法主要包括模型验证[2]与运行时验证[3]等。

基于场景的形式化验证经常使用顺序图对系统进行建模和分析,顺序图是一种基于场景的语言,能够图形化地表示系统实例间交互的时序关系。顺序图包括消息顺序图(Message Sequence Chart,简称 MSC)[4]、UML2.0 顺序图(UML2.0 Sequence Diagram,简称 UML-SD)[5]、活性顺序图(Live Sequence Chart,简称 LSC)[6]及其它变种。MSC 由 ITU 提出并在业界得到广泛应用,但 MSC 表达能力很有限,只能表示可能发生的场景,不能表示系统必须满足的场景。虽然 UML-SD 增加了一些新的操作符,增强了其表达能力,但 UML-SD 的非形式化语义也限制了其应用。LSC 对 MSC 进行扩展,LSC 使用 universal 和 existential 两种模式区分强制场景和可能场景,且 LSC 具有形式化语义。因此,软件开发过程中形式化验证经常使用 LSC 描述系统场景中事件交互的时态关系。此外,LSC 可以用来建模实际系统,也可以用于描述场景需求,基于 LSC 的形式化验证方法不仅使系统需求的行为语义易于理解,而且还能尽早地发现软件设计错误。利用 LSC 对系统进行建模和分析已成为需求分析阶段的重要技术和方法。

LSC 在 MSC 的基础上增加了活性(liveness)的概念,活性表示需求的场景一定会发生[7]。LSC 分为 universal 和 existential 两种模式,universal 图表示系统一定发生的场景,用实线矩形框表示,existential 图表示系统可能发生的场景,用虚线矩形框表示。universal 图包含一个前置图(prechart)和一个主图(main chart),前置图表示触发场景,用虚线六边形表示,主图表示响应场景,用实线矩形表示,universal 图表示的意思是,如果前置图表示的触发场景成功执行了,主图就必须执行。existential 图则只含有主图,不包含前置图,existential 图只要求前置图与主图的一个事件实例,不强制要求主图要在

收稿日期:2015-11-14; **修回日期**:2015-12-15。
基金项目:国家科技支撑计划项目(2015BAK33B00);教育部规划基金项目(15YJA880095);中央高校基本科研业务费专项资金科研项目(CCNU15GF003)。
作者简介:张 坤(1992-),湖北武汉人,硕士研究生,主要从事软件分析与验证方向的研究。

每一次前置图执行后执行。

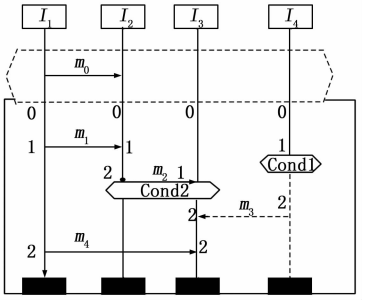

图 1　universal 模式 LSC 图

为了描述系统的强制性行为和可能性行为，LSC 对图中所有的位置、消息和条件等元素分配了温度属性，若元素的温度是 Hot，则该元素一定会发生，若元素的温度值是 Cold，则该元素可能会发生。在 LSC 图中，Hot 用实线表示，描述系统满足强制行为；Cold 用虚线表示，描述系统可能满足的行为。如果实例线上某个位置是 Hot 位置，可以直接从当前位置移动到下一个 Hot 位置；如果这个位置是 Cold 位置，实例运行可能通过该位置。如果某个消息温度值是 Hot，则该消息一定会被发送与接收；如果消息的温度是 Cold，表示该消息一定会被发送，但是消息可能被接收，也可能不被接收。

1　LSC 语言定义

下面基于文献[8]形式化定义 LSC 语言，首先定义 LSC 语法，由于基于 LSC 的形式化验证过程一般基于程序执行轨迹，这里给出 LSC 基于轨迹的语义。文献[9]论述 LSC 具有很强的表达能力，为了充分使用 LSC 进行形式化验证，有必要对 LSC 的复杂性进行论述。

1.1　LSC 语法

令 $inst(c)$ 表示 LSC 图 c 的实例线集合，$dom(c,i)$ 表示实例线 i 上的位置集合，$dom(c)$ 表示 LSC 图 c 上位置集合。

定义 1（LSC）：LSC 定义为一个 7 元组 L＝＜I，dom，ML，SR，Pre，Mode，quant＞，其中：

1）I 是实例线的结合；

2）Dom 是图 c 上的位置集合；

3）ML 是图 c 的消息集合；

4）SR 是图 c 的同步发生区域；

5）Pre 是图 c 的前置图，可为空。

6）Mode ∈ {initial，invariant，iterative}，Mode 决定 LSC 的执行频率。

7）quant ∈ {universal，existential}，quant 决定 LSC 的模式。

1.2　LSC 基于轨迹的语义

由于定义 LSC 基于轨迹的语义时要使用 LSC 的 cut，这里首先定义 LSC 的 cut。

定义 2（cut[8]）一个 cut 指图中每个实例到其位置的一个映射，$cut \subseteq dom(c,i_0)\times\ldots\times dom(c,i_n)$，其中 $inst(c)=\{i_0,\ldots,i_n\}$。

使用 cut，下面定义 LSC 的执行轨迹。

定义 3（执行轨迹）：令 cuts 序列 $c=c_0,c_1,\ldots,c_k$ 为 LSC 图 c 的一次执行，执行轨迹 $w=trace(c)$，则 $w=w_0w_1,\ldots,w_k$。其中 $c_0\xrightarrow{w_0}c_1\xrightarrow{w_1}c_2\xrightarrow{w_2}\ldots$，$w_i$ 表示图 c 中事件的集合。

一个 LSC 的轨迹语言是 LSC 图 c 执行所产生的所有执行轨迹的集合。$L_c=\{w\mid\exists(c_0,c_1,\ldots c_k)\in \mathbf{R}uns(c)\}$，使得 $w=trace(c_0,c_1,\ldots,c_k)$，其中 $Runs(c)$ 表示 LSC 所有执行轨迹的集合。

1.3　LSC 语言的表达能力与复杂性论述

在一定限制条件下，可以将 LSC 转换为等价的时序逻辑语言或者自动机语言[9]，这表明 LSC 语言具有很强的表达能力。Bontemps 等人[10]主要讨论了 LSC 语言的复杂性，表明 LSC 语言复杂性主要包括两个方面：1）LSC 语义依赖于偏序关系；2）LSC 的规约是非结构化的。前者引起的复杂性可以在实际应用中避免，因为实际应用场景一般是线性时序场景。后者引起的复杂性问题可以通过额外的信息生成高效的算法来解决。研究表明，LSC 转换为时序逻辑语言或自动机语言的复杂度相对较小，因此 LSC 经常用于形式化验证。

2　基于 LSC 的形式化验证方法

在形式化验证过程中，主要有建模系统和描述需求规约两类需求。由于 LSC 可以用来建模系统行为，也可以用于描述场景需求[8]。因此，基于 LSC 的形式化验证方法关键技术主要从以下两个方面进行论述。

2.1　使用 LSC 建模系统行为

形式化验证需要建模系统行为，MSC 可以直观的描述系统中对象间的交互情景，MSC 可以解决基于状态图作为系统行为模型的局限性问题，但是在捕获系统行为需求时，MSC 也存在一些不足[11]，LSC 也可以建模众多系统，且 LSC 语言可以补充 MSC 的不足，因此，使用 LSC 建模系统行为的研究逐渐展开。使用 LSC 建模系统行为即从基于场景规约的 LSC 构建可执行的系统。目前主要有两种方法[12]，第一种是自动合成系统的方法。给定 LSC 模型，确定是否存在一个系统满足该 LSC 模型，如果存在这样的系统，则根据 LSC 模型中实例间的交互确定实例的有限状态机或状态图，有限状态机或状态图集合构成确定满足该 LSC 的系统，但是合成的系统要满足两个需求：1）合成的系统需要监听系统中发生的事件；2）合成的系统必须满足 LSC 模型，自动合成系统的方法已在文献［11］实现。第二种方法是从 LSC 规约中场景实例间的交互建模可执行的系统[13]，该方法不是为每个实例产生实例的内部规约，而是按照算法直接执行场景，在 LSC 的场景描述中定义可执行的系统及系统运行产生的状态。这种方法有一个研究成果，即 Play-out 机制[13]，Play-out 机制允许工程师与其它相关人员更好地了解场景实例交互所产生的行为。

这两种常见的方法各有特点，两种方法的比较见表 1 所列。

2.2　使用 LSC 获取待验证的系统性质

形式化验证经常使用时序逻辑语言或自动机语言等描述系统的待验证性质。但在工程应用中，尤其是在基于场景的软件

表 1 使用 LSC 建模系统的两种方法比较

分类	主要优缺点
自动合成系统的方法	(1)建模系统效率较高,复杂度较小,处理能力较强 (2)通用性相对较弱,需要设计专门的算法
从场景实例间交互建模系统的方法	(1)通用性相对较强,有相应机制支持 (2)建模系统效率一般较低,需要优化,处理能力较弱

工程中使用时序逻辑并不实际。由于 LSC 语言具有直观性和形式化的特点，因此在形式化验证中，LSC 经常作为系统开发过程中需求规约描述语言，然后将 LSC 转换为待验证的性质，这些性质然后用时序逻辑语言或自动机语言进行描述。关于从 LSC 获取待验证的系统性质，下面分 LSC 转换为时序逻辑和 LSC 转换为自动机两种类型进行论述。

2.2.1 LSC 转换为时序逻辑

Bontemps 等人[14]对于只含 LSC 核心语法的 LSC 规约，证明任何 LSC 规约都可以转换为时序逻辑公式。Kugler 等人[15]设计算法将 LSC 转换为 LTL 公式，其它相关的研究很多基于该转换算法，文献 [15] 主要产生两种性质：图中消息的偏序性质（φ 性质），图中消息的唯一性性质（χ 性质）。对于一个 universal 图，对应的 LTL 公式 φ_c 形式如下。

$$\psi_c = G\left(\left(\begin{array}{l}\bigwedge_{p_i \prec p_j} \Phi p_i, p_j \\ \wedge \\ \bigwedge_{\forall p_i, m_j} \Phi p_i, m_j \\ \wedge \\ \bigwedge_{p_i \not\prec p_j} \neg\chi p_i, p_j\end{array}\right) \Rightarrow \left(\begin{array}{l}\bigwedge_{m_i \prec m_j} \Phi m_i, m_j \\ \wedge \\ \bigwedge_{m_j \text{ is max}} \mathbf{F}\ m_j \\ \wedge \\ \bigwedge_{\forall e_i\ m_j} \neg\chi e_i, m_j\end{array}\right)\right)$$

该转换等式从上到下分为三部分，顶部的 φ 性质保证了前置图与主图中的消息都是偏序关系，中部保证了只有前置图的消息都发生后，主图中的消息才并且一定发生，底部保证了图中的消息只发生一次。文献 [15] 中的转换方法为 LSC 图每一个可能的执行入口都生成 LTL 公式，转换后的 LTL 公式的规模是 LSC 中事件规模的平方复杂度。Kumar 等人[16]从化简偏序性性质和化简唯一性性质两个方面，利用 LSC 规约中性质的实现文献 [15] 中转换的优化，优化后所产生时序逻辑的规模是主图中最大消息规模的平方复杂度，该优化在大多数情况下可以缩小转换所产生的 LTL 公式的规模。

此外，LSC 转换为时序逻辑这些方法有一定的限制，这些方法的研究一般只针对 LSC 核心语法所表示的需求规约，缺少将完整 LSC 语法转换为时序逻辑的算法。

2.2.2 LSC 转换为自动机

文献 [17] 实现了 LSC 转换为自动机，他们将符号化自动机结构形式的 LSC 图作为输入的自动机结构，然后通过增加接受状态和增加/更新迁移将符号化自动机结构转换为可以检测安全与活性错误的否定自动机。该转换过程关注于创建自动机结构，并使用得出的自动机实现形式化验证，如果需求规约用时间扩展的 LSC 表示，则可将 LSC 转换为时间自动机。一般的转换过程关注于创建自动机结构，Kumar 等人[18]则直接研究了适用于发现系统错误的 LSC 到自动机的转换过程，他们的 LSC 转换为自动机的转换过程使得性能与可扩展性均显著提升。

将 LSC 规约转换为自动机语言，然后使用基于自动机语言进行形式化验证，这种方法可以支持 LSC 的更大语法子集，且支持更大规模的 LSC 规约[17]。

3 基于 LSC 的形式化验证典型应用

基于 LSC 的形式化验证应用已有很多，从形式化验证方法分类来看，目前基于 LSC 的形式化验证主要应用有模型验证领域与运行时验证领域等。下面就基于 LSC 的形式化验证在模型验证领域与运行时验证领域的应用进展加以论述。

3.1 基于 LSC 形式化方法的模型检测

LSC 可以作为系统需求规约描述语言，通过将 LSC 转换为时序逻辑语言或自动机语言的方法可以从 LSC 获取系统待验证的性质[17]。此外，LSC 又可以建模系统行为，因此，LSC 也是系统的高级编程语言。模型检测即需要建模系统也需要时序逻辑或自动机语言等表示的需求规约，而 LSC 正好具有上述两种表述能力。因此，很多模型检验应用研究都基于 LSC，基于 LSC 形式化方法的模型检验应用研究可以分为三类：

1）LSC 只转化为系统行为模型，待验证的系统性质规约由用户定义[17]；

2）系统行为模型由其它模型产生，待验证系统性质规约从 LSC 中获取[19]；

3）系统行为模型由 LSC 转换形成，待验证系统性质规约也从 LSC 中获取[8]。

其中，情形 3）的基于 LSC 形式化方法的模型检测过程如图 2 所示。

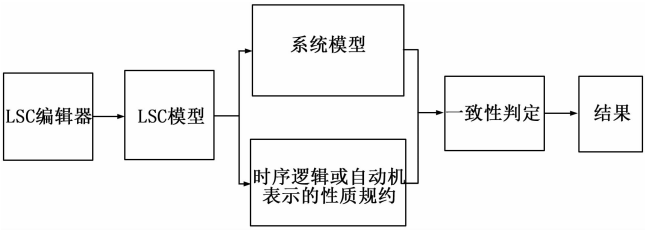

图 2 基于 LSC 形式化方法的模型检测过程

3.2 基于 LSC 形式化方法的运行时验证

LSC 可以转换为时序逻辑语言或者自动机语言来获取系统需求规约，将转换的时序逻辑语言或者自动机语言表示的需求规约作为监控器[3]，使用运行时验证的相关技术[20]就可以实现基于 LSC 形式化方法的运行时验证。基于 LSC 形式化方法的运行时验证过程如图 3 所示。

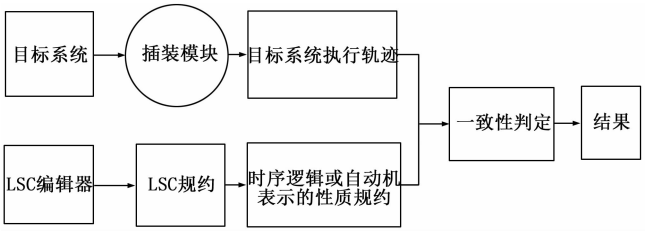

图 3 基于 LSC 形式化方法的运行时验证过程

基于 LSC 的形式化方法应用于运行时验证领域已经有相关研究，Chai 等人[21]提出中标准 LSC 语言在其研究的嵌入式

实时系统中表示特定场景的正确性性质还不够充分，因此引入充分前置图的概念，提出扩展的活性顺序图（extension of live sequence charts，简称 eLSC），并分为有迭代 eLSC 和无迭代 eLSC 两种场景，有迭代 eLSC 场景直接使用文献[15]中的方法转换成为 LTL 公式，并使用公式重写的方法使用 Maude 工具引擎[22]进行运行时验证，无迭代 eLSC 场景则设置专门的算法进行运行时验证。并研究了欧洲列车控制系统（ETCS）中 RBC/RBC 切换场景，实现基于 LSC 形式化方法用于运行时验证。

4　基于 LSC 的验证工具实现

本文实现的基于 LSC 的验证工具的设计思路是：首先，插装获取系统执行轨迹，导入目标系统程序，使用 Spring AOP 进行代码插装获取其执行轨迹；然后，输入使用从 LSC 转换所得的 LTL 公式；最后，自动生成被验证的项目文件，再调用 Maude 工具引擎实现运行时验证。

4.1　工具架构

本文设计的基于 LSC 的验证工具架构如图 4 所示。

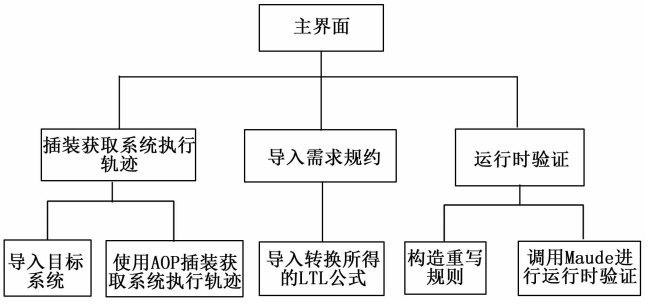

图 4　运行时验证工具框架

1）插装获取系统执行轨迹：

在此模块中，用户可以导入要验证的目标系统源程序，并且可以在工具中查找已导入的目标系统程序代码。在目标系统程序运行时，根据配置的插装点通过 Spring AOP 插装获取目标系统的执行轨迹序列，并将执行轨迹序列存贮在本地，便于后续的运行时验证。

2）导入需求规约：

在此模块中，根据 LSC 描述的需求规约场景，用户可以导入从 LSC 转换所得的 LTL 公式，导入的 LTL 公式作为需求规约将存贮在本地，便于后续的运行时验证。

3）运行时验证：

在此模块中，工具根据公式重写算法自动生成相应的重写规则，Maude 工具引擎根据重写规则文件运行产生监控器，监控器分别根据（1）和（2）获取的系统执行轨迹和需求规约进行重写逻辑的运行时验证，并显示验证结果。

4.2　工具实现

4.2.1　插装获取系统执行轨迹模块实现

该模块的作用是通过 Spring AOP 插装获取目标系统的执行轨迹。在该模块中，GetSystemTrace 类调用目标系统运行，RBC2RBCAspect 类根据配置的插装点进行代码插装，当目标系统运行完成后，RBC2RBCAspect 类的通知代码获取系统的执行轨迹序列。然后由 GetSystemTrace 类生成原子命题声明文件与执行轨迹序列日志文件。如图 5 所示是本模块的静态结构图。

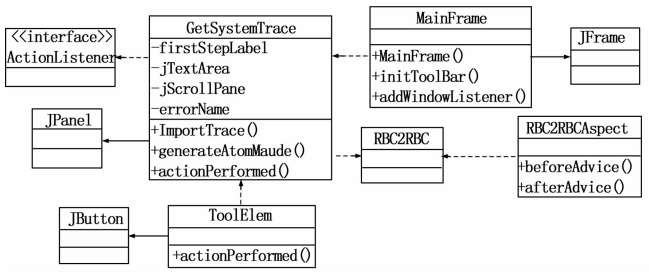

图 5　插装获取系统执行轨迹模块静态结构图

4.2.2　需求规约导入模块实现

该模块的作用是导入从 LSC 转换所得的 LTL 公式，同时生成相应的需求规约文件并将其存储在本地，生成的需求规约文件将作为需求规约输入到 Maude 工具引擎中。如图 6 所示是本模块的静态结构图。

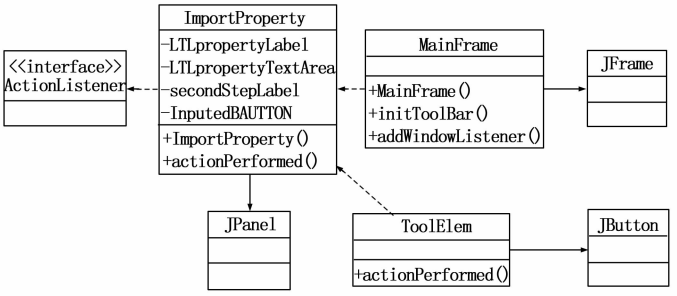

图 6　需求规约导入模块静态结构图

4.2.3　运行时验证模块实现

该模块的作用是调用 Maude 工具引擎进行运行时验证。该模块中，RunDemo 类生成相应的重写规则，Maude 工具引擎根据重写规则文件进行运行时验证，并显示运行时验证结果。如图 7 所示是本模块的静态结构图。

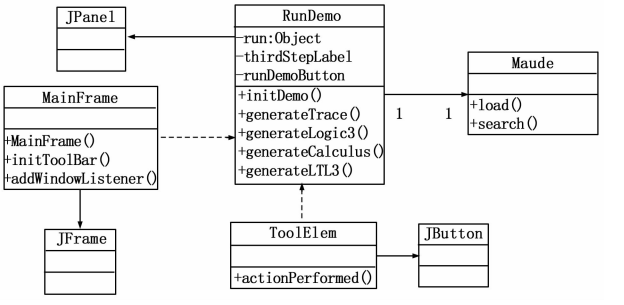

图 7　运行时验证模块的静态结构图

4.2.4　验证过程

第一，插装获取系统执行轨迹。首先，工具将导入并运行目标系统，当目标系统运行结束后，工具就获取了执行轨迹序列。第二，导入待验证的需求规约，输入从 LSC 转换所得的 LTL 公式形式的需求规约。最后，进行运行时验证。工具首先生成重写规则文件，然后调用 Maude 工具引擎，Maude 工具引擎根据重写规则产生预测监控器，进行运行时验证，并显示验证结果。本次实验结果如图 8 所示。

本次实验结果为“maybe”，由于从 LSC 转换所得的 LTL 公式含有时序逻辑运算符 G，验证工具分析目前的执行轨迹没有发生需求违约，但对以后的执行情况还无法确定，判定结果

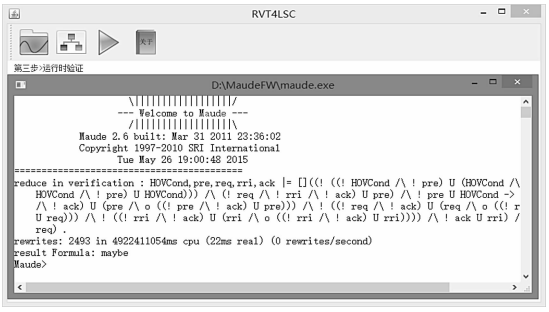

图 8 运行时验证

为可能为真，因此验证结果为 maybe，这也体现出 LTL 三值语义实现了对 LTL 二值语义的自然覆盖。然后进行另一组实验，本次插装获取另一条轨迹 t = HOVCond，pre，req，ack，rri，使用工具进行运行时验证，该执行轨迹的实验结果为“false”，如图 9 所示。

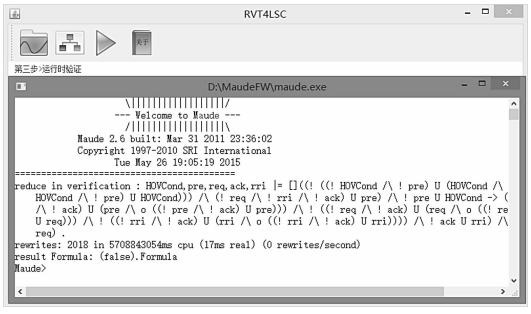

图 9 执行轨迹违反规约的验证结果

5 结语

基于 LSC 的形式化验证方法已成为软件开发过程中一种重要的验证方法，基于 LSC 的形式化验证已经成功应用到多个领域。本文对基于 LSC 的形式化验证方法进行研究，论述 LSC 的理论基础，对比分析了基于 LSC 的形式化验证领域的关键技术与应用进展，并实现一种基于活性顺序图的验证工具，实验证明使用本验证工具进行运行时验证的可行性。随着软件规模不断扩大系统集成度不断提高，基于 LSC 的形式化验证方法必将在软件开发过程中发挥其应有的价值。

参考文献：

[1] Li X S, Liu Z M, He J F. A formal semantics of UML sequence diagram [A]. Proceedings of the 2004 Australian Software Engineering Conference [C]. 2004: 168 - 177.

[2] Clarke E M, Grumberg O, Peled D A. Model Checking [M]. MIT press, 1999.

[3] Leucker M, Schallhart C. A brief account of runtime verification [J]. The Journal of Logic and Algebraic Programming, 2009, 78 (5): 293 - 303.

[4] ITU-T. Recommendation Z. 120: Message Sequence Charts [Z]. Geneva: ITU-T, 1999.

[5] Object Management Group (OMG). UML [Z]: Superstructure Version 2. 2. 2009.

[6] Damm W, Harel D. LSCs: Breathing Life into Message Sequence Charts [J]. Formal Methods in System Design, 2001, 19 (1): 45 - 80.

[7] 李雯睿，王志坚，张鹏程. 模态顺序图 uMSD 的形式化语义 [J]. 软件学报，2011，22 (4)：659 - 675.

[8] Larsen KG, Li S. Scenario-based analysis and synthesis of real-time systems using Uppaal [A]. Proc13th Conference on Design, Automation, and Test in Europe [C]. 2010: 447 - 452.

[9] Harel D, Maoz S, Segall I. Some Results on the Expressive Power and Complexity of LSCs [J]. In Pillars of computer science, 2008, 351 - 366.

[10] Bontemps Y, Schobbens P. The Complexity of Live Sequence Charts [J]. Institut d' Informatique, University of Namurrue Grandgagnage, 21B5000-Namur (Belgium), 2005, 15.

[11] Harel D, Segall I. Synthesis from scenario-based specifications [J]. Journal of Computer and System Sciences, 2012: 78 (3), 970 - 980.

[12] Harel D, Kantor A, Maoz S. On the Power of Play-Out for Scenario-Based Programs. [J]. Lecture Notes in Computer Science, 2010: 207 - 220.

[13] Brenner C, Greenyer J, Panzica V et al. The ScenarioTools Play-Out of Modal Sequence Diagram Specifications with Environment Assumptions [A]. In Proc. 12th Int. Workshop on Graph Transformation and Visual Modeling Techniques. Volume 58, EASST [C]. 2013.

[14] Bontemps Y, Schobbens P. The Computational Complexity of Scenario-based Agent Verification and Design [J]. Journal of Applied Logic, 2007, 5 (2): 252 - 276.

[15] Kugler H, Harel D, Pnueli A et al. Temporal logic for scenario-based specifications [A]. Proc. of the 11th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS05) [C]. 2005: 3440, 445 - 460.

[16] Kumar R, Mercer E, Bunker A. Improving translation of live sequence charts to temporal logic [A]. Proc. of the 7th Int. Conf. on Automated Verification of Critical Systems (AVoCS07) [C]. 2007, 183 - 197.

[17] Kumar R, Eric G Mercer. Verifying Communication Protocols Using Live Sequence Chart Specifications [J]. Electronic Notes in Theoretical Computer Science, 2009, 250 (2): 33 - 48.

[18] Kumar R, Mercer EG. Improved live sequence chart to automata translation for verification [J]. Electronic Communications of the EASST, Volume 10, 2008.

[19] Li SH, Balaguer S, David A, G. Larsen K, Nielsen B, Pusinskas S. Scenario-based verification of real-time systems using UPPAAL [C]. Form Methods System Design, 2010, 37: 200 - 264.

[20] 张 硕，贺 飞. 运行时验证技术的研究进展 [J]. 计算机科学，2014，41 (11A)：359 - 363.

[21] Chai M, Schlingloff B. Monitoring Systems with Extended Live Sequence Charts [J]. Springer International Publishing Switzerland, RV 2014, LNCS 8734, 48 - 63.

[22] Clavel, Duran, Marti-Oliet, Meseguer, Talcott et al. Maude Manual [M] (version 2. 6). University of Illinois, Urbana-Champaign, 2011.

文章编号:1671-4598(2016)05-0279-05 DOI:10.16526/j.cnki.11—4762/tp.2016.05.077 中图分类号:TP3 文献标识码:A

系统健康管理中的寿命模型仿真研究

李文娟，刘海强
（西安科技大学 通信与信息工程学院，西安 710054）

摘要：在航空、航天、通信等领域，高可靠性和长寿命设计的产品所占比重逐渐增大；性能退化状态评估和剩余使用寿命预测技术在提高该类产品安全性和维护效率、降低全寿命周期成本等方面意义重大；针对当前国内健康管理研究中缺乏寿命及可靠性基础数据的现状，介绍了系统寿命预测的典型过程，重点分析了寿命模型研究和仿真中存在的若干问题，旨在建立实现简单且与实际系统运行过程相似度高的寿命模型的建模方法，为后续系统性能状态评估提供有效的数据支持。

关键词：性能退化；寿命模型；系统仿真

Research on Life Model Simulation of the System Health Management

Li Wenjuan，Liu Haiqiang
（Communication and Information Engineering College，Xi'an University of Science and Technology，Xi'an 710054，China）

Abstract：In the areas such as aviation，aerospace and communication，high reliability and long life design products are widely used. Assess of the performance degradation status or prediction of the remaining useful life have great significance in enhancing system safety，increasing maintenance efficiency and reducing total life cycle cost of the system. To cover the shortage of life and reliability data in domestic health management，the system representative modeling process is presented. Especially，the existent problems for further research of life model and modeling simulation are investigated，which is to obtain the life model easy to fulfill and with high-fidelity. Then it can be hoped to provide efficient data support for further performance degradation status assessment.

Keywords：performance degradation；life model；system simulation

0 引言

科学技术的发展和工业水平的提高，使得许多高可靠、长寿命产品被广泛地应用于航空、航天、通信等高新技术领域。随着产品的设计越来越精密，结构越来越复杂，出现可靠性隐患和发生故障的几率也呈现不断增长的趋势，直接导致产品的寿命缩短以及维护费用增大。

对于性能退化型产品，随着使用时间的退化，通过对这些特征参数进行跟踪[1]，能增加，表征产品特性的性能参数往往会出现够对产品进行故障的预测和规避。故障预测与健康管理PHM（prognostics and health management）中的预测是指以系统历史和当前状态为依据，以一定的可靠性为约束条件，通过设定失效阈值标准和建立产品寿命分布模型，推测出系统未来发生故障的时间、类型和程度。有效的PHM方法在大幅度降低设备全寿命周期成本的同时，能有效降低安全事故的发生概率，对于产品在各领域的应用具有重要意义。

近年来，国内外学者在系统寿命预测领域开展了比较深入地研究。Bhaskar Saha等[2]等以电池为对象，分析了其失效原因，选取电压、电流、功率等作为主要性能参数，构建了其性能退化模型；周月阁等[3]以某型装备上的恒流电源为例，实现了基于性能退化和Monte-Carlo仿真的系统性能可靠性评估过程，验证了该方法的有效性。赵建印等[4]提出了竞争失效模型，给出了两种基于参数回归模型的竞争失效统计方法。王玉明等[5]提出了基于多元概率密度函数和主成分分析的多特征量退化数据可靠性建模方法，使得多个相关退化量转化为少数几个不相关的综合特征量。

通过对比国内外研究现状可以发现：首先，由于现代系统自身的复杂性，建模与仿真可以作为解决其剩余寿命预测中数据缺乏问题的有效途径；其次，目前系统寿命模型相对较少，现有模型对环境、使用因素的考虑比较欠缺，不利于工程应用。因此，需要对系统建模作更为细致的完善工作，尽可能建立准确详尽而又适于仿真的系统仿真模型；另外，针对性能退化系统工程的应用需求，不断开展先进的建模方法，加强寿命模型的可验证性，才能有效提高系统剩余寿命预测技术的高效性和实用性。

1 关于寿命模型

1.1 系统的健康状态与剩余使用寿命

失效轨迹是指系统从功能完好到失效所经历的路径，它描述了系统健康状态的变化情况。产品典型的失效轨迹如图1所示，曲线表示系统的性能退化曲线。D点为系统发生故障的时间点。从A点开始，系统进入功能受损状态，其中，A～C段为关键监测时间段，B～C段为关键预测时间段[6]。

故障预测指的是预测未来时间段内故障第一次出现的时间，而剩余使用寿命（remaining useful life，RUL）是预测产品的总寿命，在该时间段内尽管会出现故障和维护行为，系统的功能能够实现。

1.2 性能特征参数和健康指标

系统必须保证有效地实施其预定功能，否则认为其发生性能退化或出现故障，而故障征兆常常表现为其运行参数或性能参数的变化。因此，能够表征产品性能明显退化或是与设备性能具有敏感性的参数被选定为性能退化特征输出量。

系统的性能退化状态量可以用健康指数（Health Index，

收稿日期：2015-10-12； **修回日期**：2015-12-18。
基金项目：学校基金(2015QDJ045)。
作者简介：李文娟(1979-)，女，陕西扶风人，博士，讲师，主要从事故障与诊断控制、健康管理技术等方向的研究。

HI）或退化指数（Degradation Index，DI）[7]来表示，用以描述系统的健康状态。DI 与维护及后勤行为的对应关系如表 1 所示。例如当 DI=0.7 时，对应系统的操作能力：功能严重受损；需要的维护行为：定期更换；需要的后勤行为：紧急采购[6]。

表 1 定义退化指数对应的维护行为

DI	操作能力	维护行为	后勤行为
[0～0.2]	功能完好	无维护行为	无需改变后勤计划
[0.2～0.4]	功能满足但伴随性能降级	方便时维护	方便时采购
[0.4～0.6]	功能缩减	立即维护	需要采购
[0.6～0.8]	功能严重受损	定期更换	紧急采购
[0.8～1.0]	失效	立即更换	元件报废

例如，绕组短路是电机性能退化的一种常见模式，引起电机扭矩的变化，从而导致电机电流等参数的变化。因此，可以通过分析电流来判断系统内是否出现性能退化现象。选取电机电流作为系统的特征输出量，选取绕组未短路匝数作为性能退化状态量即通过仿真电机电流，获取系统的性能退化状态量绕组短路匝数。

2 基于寿命模型仿真的 RUL 预测

典型的基于仿真模型的寿命预测如图 1 所示[8]，具体包含以下过程。

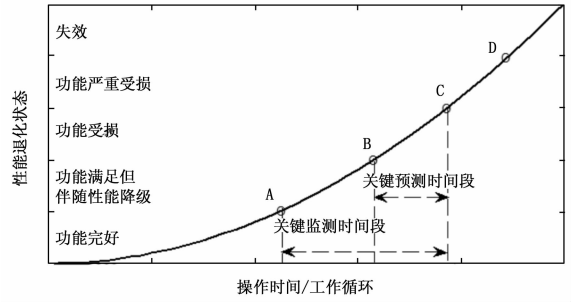

图 1 产品失效轨迹

2.1 失效模式和机理的选择

首先，使用可靠性工具，例如故障模式、机理及影响分析（Failure Modes，Mechanisms，and Effects Analysis，FMMEA），计算产品失效模式和机理的风险优先数。

由于失效机理检测的困难性，预先对 FMMEA 和环境条件进行评估，确定被监测失效征兆的概率，以确保收集适于预测的数据，即基于 FMMEA 的失效模式和影响分析，可以选择重要性级别最高且适于模拟的失效模式和机理，一定程度上也能够优化预测成本。

2.2 寿命模型的建立

分析失效模式产生的退化机理，建立功能完整、实现简单且与实际系统运行过程相似度高的系统寿命模型，能够为后续性能状态评估提供有效的数据支持。

为了验证所选取性能退化关键元器件及其失效模式的有效性，通过对性能退化相关参数灵敏度仿真，分析其扰动变化时系统模块性能退化输出特征量的变化情况，从而可以对性能退化模式的选取进行修正。将经过验证后的退化轨迹注入到仿真模型中，获得系统主要性能参数由于性能退化关键元器件的退化而出现的变化规律。

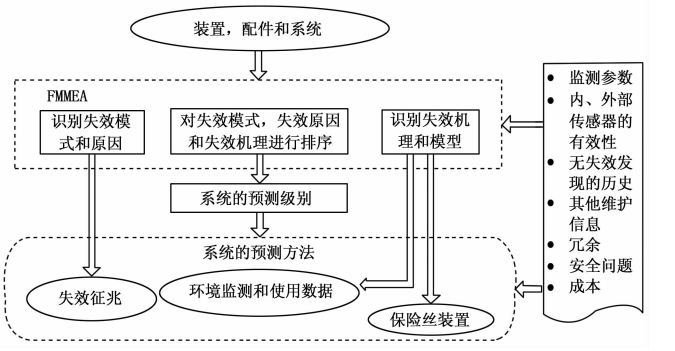

图 2 系统预测实施过程

2.3 预测及决策算法的开发

预测推理技术以系统健康数据为基础，开发通用且易于验证的预测算法。预测方法大致可以分成以下几种[9]：基于经验的方法，例如采用剩余使用寿命的失效概率密度函数进行预测；基于趋势模型的方法，例如采用数据驱动或特征关联的方法进行预测，如模糊逻辑、神经网络等；第 3 种方法为基于物理模型的方法，例如采用系统的失效机理模型进行预测。

决策支持是健康管理的最终结果，即在健康评估和故障预测的基础上，结合各种资源，进行维修决策的自动生成、维修资源的统一调配以及各相关单位的协同保障等。

3 寿命模型建立过程中存在的若干问题

3.1 多工作模式损伤累积模型

频繁的周期性操作和多工作模式是复杂系统运行的基本特点，即系统运行时会经历一系列从开启到关闭的周期性连续循环过程。每一个工作循环中，系统的行为模式是变化的，对应各模式下的性能退化情况也不同。研究系统的性能退化规律，就需要对各行为模式区别对待，然后计量到一个完整的工作循环。

假设系统的性能退化状态由 l 个子系统的性能状态表征，令 x_k^i 表示第 k 个循环结束时，系统中第 i 个子系统对应的性能退化状态量。x_k^i 的变化取决于以下两个因素：1）第 i 个子系统在第 $k-1$ 个循环结束时的状态 x_{k-1}^i；2）第 i 个子系统在第 k 个工作循环内每一个行为模式下的性能退化函数。建立性能退化累积函数如式（1）～（2）所示[10]。

$$x_k^i = Ax_{k-1}^i - B\sum_{j=1}^{m} x^i(\theta_j^i(\tau), u_j^i(\tau))$$
$$k = 1,2,\ldots,n; \quad i = 1,2,\cdots l \tag{1}$$

$$\frac{d\theta_j^i}{d\tau} = \Psi(\theta_j^i(\tau), u_j^i(\tau)), \quad 0 \leqslant \tau \leqslant \Delta t \tag{2}$$

其中，n 和 m 分别表示工作循环数和每个工作循环内行为模式（持续时间为 Δt）的个数。$\frac{d\theta_j^i}{d\tau}$ 和 $u_j^i(\tau)$ 分别表示每个工作循环内第 j 个行为模式下子系统 i 的性能退化参数和特征输出量的变化量。

多工作模式系统寿命模型建模方案如图 3 所示。其步骤可总结如下：①选取系统中实际存在且适于模拟的性能退化模式，并分析其退化机理，建立对应的性能退化数学模型；②分析系统各工作循环内，性能退化可能出现的行为模式及行为模式的组合；③设定系统的性能退化阈值，通过循环累积效应来模拟性能退化过程。

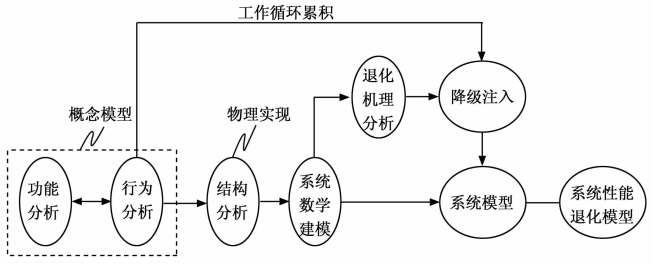

图 3　多工作模式系统寿命模型建模方案

3.2　多性能参数寿命模型

现代产品功能众多，工作环境复杂多变，使得系统性能退化进程间存在着相互关系。建立在独立性假设基础上的传统可靠性模型，用简单的加速因子或环境因子，无法刻画出多种因素及其复杂相关性对产品可靠性的影响，导致可靠性预计值和实际情况相差较大。

系统的性能降级需要将潜在失效模式与目标元件故障进行关联，从而实现有效诊断和预测。关联过程通过只保留与故障相关的信号，即目标系统失效 BIT[11] 来过滤系统信息，产生一个与目标系统失效最为相关的简化参数集。

通过典型的 BIT 集，识别状态指示量的过程如图 4 所示，简化连接矩阵示意了不同子系统之间的互联性和相关性，实现对不同子系统之间诊断模糊性的缩减。例如，目标系统与子系统 1、2、3 都具有关联关系。通过 BIT6 识别目标系统和子系统 3 之间的关系，通过 BIT7 识别目标系统和子系统 4 之间的关系，通过 BIT4 识别目标系统和子系统 1 之间的关系。状态指示量确定以后，对其进行追踪或趋势分析，可以用来指示系统的异常行为。

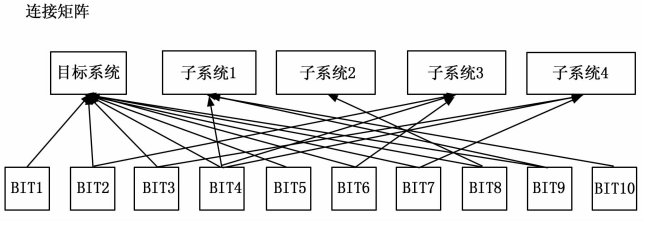

图 4　精简的连接矩阵

3.3　间歇性失效模型

系统（特别是电子产品）的许多失效在本质上都是间歇性发生的，即系统在一定的时间段内一些功能或性能特性有所损失，该时间段结束后，功能又重新恢复。间歇性失效检测、识别和重现性比较困难。

Zhang Guangfan 等结合环境负载（比如温度）和现场性能/工作参数的测量值，融合两个基本预测算法：寿命消耗监测和不确定性调整预测，提出了一个电子系统 RUL 增强预测模型，如图 5 所示[12]，并以温度循环负载下，用间歇和“硬”焊接接头互联设备的失效来验证了该预测模型除了监测“硬”故障，还有评估间歇故障的能力。

3.4　子系统健康模型融合

系统健康状态表现为其子系统性能上的降级、关键设备冗余级别的变化等。子系统的预测评估结果如何用于系统的维护和后勤决策，是 PHM 的一个关键问题。

系统的 RUL 用日历时间或工作循环来表示，最小剩余使

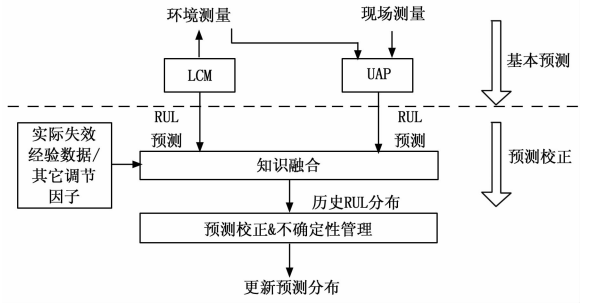

图 5　增强型预测模型

用寿命的思想将系统的 *RUL* 用其 l 个重要子系统的剩余使用寿命的最小值来描述[13]。该思想可描述如下：

令 t_p 表示当前给定的预测时间点；EOL_k 表示子系统 k，($k=1,2,\cdots l$) 的寿命终止时刻；T_{EOL_k} 表示子系统 k 性能退化的判定阈值。T_{EOL_k} 的定义如式（3）所示。

$$T_{EOL_k}(x_k(t),\theta_k(t)) = \begin{cases} 1 & t \geqslant EOL_k \\ 0 & t < EOL_k \end{cases} \tag{3}$$

其中，$x_k(t)$ 和 $\theta_k(t)$ 分别表示 t 时刻子系统 k 的性能退化特征量和性能退化指数。式 3 表示：如果通过参数 $x_k(t)$ 和 $\theta_k(t)$，判定子系统 k 的已用时间 t 到达其寿命终止时刻 EOL_k，则认为子系统 k 已经不能满足其功能要求，令 $T_{EOL_k}=1$；否则令 $T_{EOL_k}=0$。

系统的寿命终止时刻 *EOL* 由首先满足寿命结束条件（$T_{EOL_k}(x_k(t),\theta_k(t))=1$）的 EOL_k 来确定，则 t_p 时刻系统的剩余使用寿命（$RUL(t_p)$）等于 *EOL* 与 t_p 的差值，具体定义如式（4）所示。

$$\begin{aligned} EOL &\overset{\Delta}{=} \underset{t \geqslant t_p}{\operatorname{argmin}} T_{EOL_k}(x_k(t),\theta_k(t)) = 1 \\ RUL(t_p) &= EOL - t_p \end{aligned} \tag{4}$$

3.5　寿命消耗监控模型

产品的生产、装运、存储、处理、使用等过程中往往会产生寿命周期负载，如表 2 所示[14]。产品所经历的热、机械学、化学和电学等，分别或以不同的组合导致产品性能退化，缩减其服役寿命。例如，在应力 & 损伤预测中，产品退化的程度和比率取决于其暴露至负载（使用率、频率和强度）的强度量级和持续时间。

表 2　寿命负载示例

负载	负载条件
热学	稳态温度，温度范围，温度循环，温度梯度，升温速率，散热
机械	压力幅度，压力梯度，振动，冲击负载，声级，应变，应力
化学	湿度梯度，大气等环境污染，臭氧，引起生物有害反应的污染，燃料溢出
物理	辐射，电磁干扰，海拔高度
电学	电流、电压、功率

将这些可测量的负载剖面和损伤模型相结合，使用寿命模型、材料属性、设计定义以及寿命周期环境剖面信息等建立寿命消耗监控（LCM，Life Consumption Monitoring）模型[15]，能评估由于累积的负载暴露引起的性能退化，如图 6 所示。

3.6　寿命周期成本模型

故障预测的目的之一是缩减成本，包括减少后勤和维护成本，然而，预测实施的过程本身就会增加成本。系统的寿命周

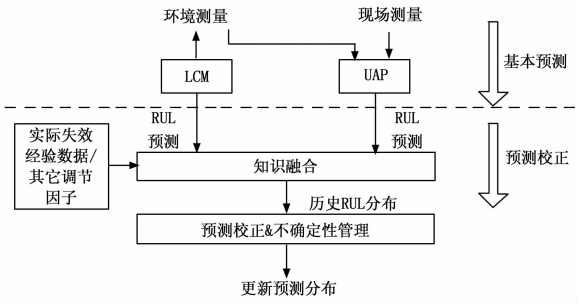

图 6 基于寿命消耗监控模型的 RUL 预测

期成本（Life Cycle Cost，LCC）是指某一系统从概念设计、技术设计、生产制造、使用维护，直至失效报废或回收再利用的整个期间所产生的直接或间接的成本总和。

寿命周期成本模型的一个重要目的是研究如何使用信息去最有效地管理设备的使用、维护和后勤，为待研究产品的 PHM 提供可论证的技术方案。

N. B. Hölzel 通过对飞机的操作和维护成本进行建模如图 7 所示，描述了飞机寿命周期成本的分析及评估过程[16]。

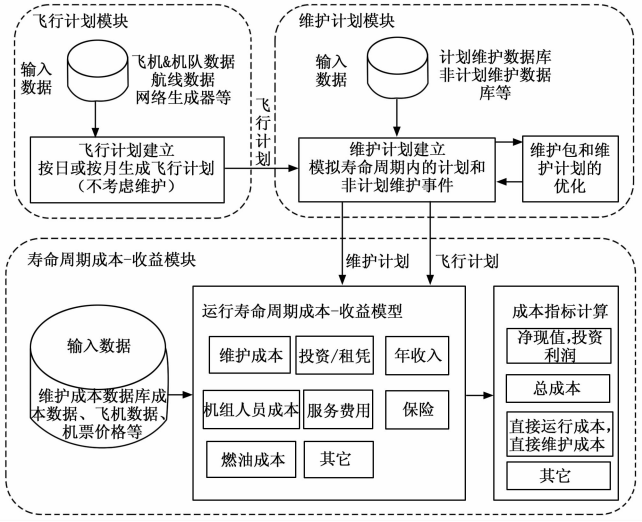

图 7 飞机寿命周期成本－收益模型概述

3.7 考虑不确定性的寿命模型

基于失效机理（Physics-of-Failure，PoF）电子预测的部分不确定源如图 8 所示，大概可以分为 3 类[17]：①由模型简化和参数不精确引起的模型不确定性；②由环境和操作负载引起的测量和预测不确定性。③由生产过程所引起产品特性参数的不确定性，例如，负载压力分析模型中的不确定性可能来源于材料与几何参数的变化；失效疲劳模型的不确定性可能来源于疲劳常数的波动等。

上述不确定性因素能导致预测结果与实际寿命具有比较大的误差，甚至出现错误的预测结果，例如不同负载（低负载、正常负载和高负载）条件下产品的 RUL 预测结果有所不同。建立包含不确定性的有效寿命模型，才能为 PHM 结果提供有用的信息。

考虑不确定性分析的预测模型如图 9 所示[18]。通过敏感性分析来选择不确定性的关键参数，并为其分配合适的概率密度函数；并使用蒙特卡洛模拟方法来随机采样，从损伤累积分布出发，预测带置信度间隔的剩余寿命，用来评估预测不确定性和参数不确定性如何影响预测结果。

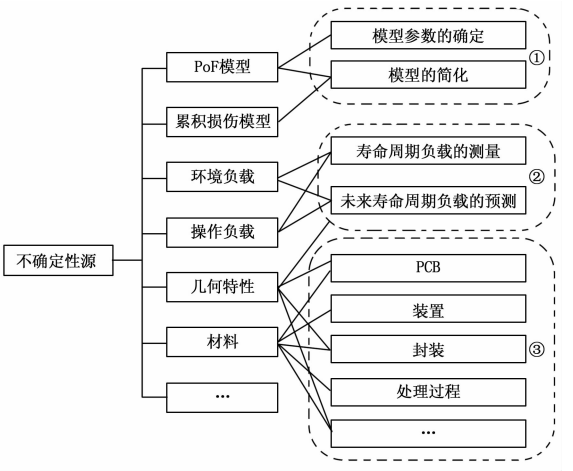

图 8 基于 PoF 电子寿命预测的部分不确定源

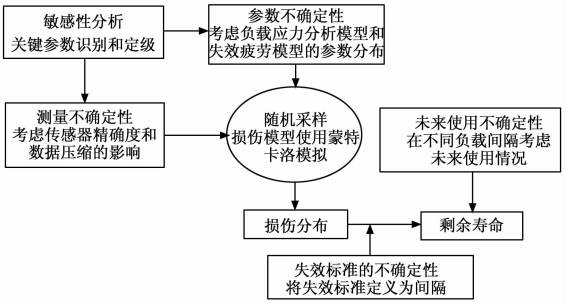

图 9 预测的不确定性分析

4 结论

综上所述，系统寿命模型的建立涉及领域广泛，需要多学科交叉、创新、积累。国内外关于此类领域的研究刚刚起步，资料比较匮乏，基本停留在技术探讨阶段，还未形成有效、实用的方法。对已有成果进行深入研究的基础上，探讨一些有效的改进方法用于解决系统寿命模型仿真研究等关键问题，继而发展系统级设备的综合性能状态评估及后勤决策支持，能够有效提高系统的可靠性及维护效率。

参考文献：

[1] Tuchband Brian A. Implementation of Prognostics and Health Management for Electronic Systems [D]. Park：University of Maryland. 2007.

[2] Bhaskar Saha. A Bayesian Framework for Remaining Useful Life Estimation [J]. Association for the Advancement of Artificial Intelligence. 2007.

[3] 周月阁. 基于性能退化和 Monte-Carlo 仿真的系统性能可靠性评估 [J]. 仪器仪表学报. 2014. 35 (5)：1185－1191.

[4] 赵建印. 基于性能退化数据的可靠性建模与应用研究 [D]. 长沙：国防科学技术大学，2005.

[5] 王玉明. 基于性能退化数据的电子产品可靠性分析研究 [D]. 石家庄：军械工程学院博士论文，2009.

[6] Patrick W. Kalgren. Defining PHM，A Lexical Evolution of Maintenance and Logistics [A]. IEEE Autotestcon [C]. Anaheim，CA. 2006. 9. 18－21：357.

[7] Van Tung Trana. Machine Performance Degradation Assessment and Remaining Useful Prediction using Proportional Hazard Modeling and Support Vector Machine [J]. Mechanical Systems and Signal Processing, 2012, 32: 320 - 330.
[8] Gu J. Prognostics Implementation Methods for Electronics [A]. Reliability and Maintainability Symposium. Annual [C]. 2007. 1. Orlando: 101 - 106.
[9] Michael J. Roemer. An Overview of Selected Prognostic Technologies with Application to Engine Health Management [A]. ASME Turbo Expo [C]. Barcelona, Spain. 2006. 5. 8 - 11: 2.
[10] Pradeep Shetty. A Hybrid Prognostic Model Formulation System Identification and Health Estimation of Auxiliary Power Units [A]. IEEE Aerospace Conference [C]. Big Sky, MT. 2006: 2 - 5.
[11] Philip L. Dussault. Field Data Evaluation and Continuous Health Assessment of Critical Avionics Subsystem Degradation [A]. Aerospace Conference [C]. IEEE, 2006: 1 - 8.
[12] Zhang GF. An Enhanced Prognostic Model for Intermittent Failures in Digital Electronics [A]. Proceedings of the 2007 IEEE aerospace conference [C]. 1 - 8.
[13] Matthew Daigle. Distributed Damage Estimation for Prognostics based on Structural Model Decomposition [A]. Annual Conference of the Prognostics and Health Management Society [C]. 2011: 2.
[14] Nikhil M. Vichare. Prognostics and Health Management of Electronics [J]. IEEE 2009. 3: 222 - 229.
[15] Chris Wilkinson. Prognostic and Health Management for Avionics [J]. IEEE. 2004: 1 - 13.
[16] Hölzel N. B. System Analysis of Prognostics and Health Management Systems for Future Transport Aircraft [A]. 28th International Congress of the Aeronautical Sciences [C]. 2012: 1 - 13.
[17] Sun B. Benefits Analysis of Prognostics in Systems [A]. Prognostics & System Health Management Conference. Macau [C]. 2010. 12.
[18] Gu J. Uncertainty Assessment of Prognostics of Electronics Subject to Random Vibration [A]. AAAI fall symposium on artificial intelligence for prognostics [C]. 2007. 11: 50 - 7.

(上接第 273 页)

表 3　11 楼 20150721 多普勒频移计算

SVN	实测多普勒/Hz	理论多普勒/Hz	实测与理论的差值/Hz	修正晶振误差后的差值/Hz
3	－1923	－2219	296	－3
6	1732	1434	298	－1
17	579	280	299	－0
23	2222	1925	297	－2
28	－2664	－2958	294	－5

注：晶振误差：－0.19ppm，即－0.19 * 10－6 * 1575.42 * 106＝－299 Hz 的频率偏差。

可以看到通过修正晶振误差后，5 颗卫星的实测与理论多普勒频移差值在－3 Hz 左右。

表 4　11 楼 20150909 多普勒计算

SVN	实测多普勒/Hz	理论多普勒/Hz	实测与理论的差值/Hz	修正晶振误差后的差值/Hz
2	1552	873	678	1
5	2577	1895	681	4
6	－655	－1329	674	－3
9	－764	－1442	678	1
12	－1521	－2200	679	2

注：晶振误差：－0.43ppm，即－0.43 * 10－6 * 1575.42 * 106＝－677 Hz 的频率偏差。

可以看到通过修正晶振误差后，5 颗卫星的实测与理论多普勒频移差值在 3 Hz 左右。

由以上 4 组数据的理论计算和实测数据的对比可以得出，任意用户位置的多普勒频移理论计算符合实测数据的多普勒频移。11 楼 2 组数据的多普勒频移误差完全是由晶体振荡误差引起，而 2 楼 2 组数据仍存在其他误差来源，可能来源是：(1) 2 楼位置是由 11 楼推算出来的，可能存在位置误差；(2) 2 楼的采集位置处离周围墙壁较近，可能存在多径。

4　结论

本文提出了在地心地固坐标系下任意用户位置载波多普勒频移最大值的精确计算方法，考虑地球自转以及用户位置对多普勒频移的影响，修正了之前 TSUI 的关于多普勒频移最大值的推导；并且通过实验验证其理论计算曲线基本符合实际测试曲线。

本文同时提出了一种基于精密星历的任意用户位置载波多普勒频移计算方法，可以计算出任意用户位置任意时刻（精密星历时间范围内）的多普勒频移，同时也通过实测数据进行验证，其理论计算值和实测值在修正晶振误差后基本相等（误差在 20 Hz 以内）。本文提出的载波多普勒频移计算方法在已知精密星历和用户位置情况下则可以计算结果，更易于计算出多普勒频移；并且此计算方法可以得到任意用户位置任意时刻的载波多普勒，在实测数据难以得到时可以代替实测数据计算多普勒频移，所以此计算方法对多普勒频移的理论研究非常有意义。

参考文献：

[1] 许晓勇，王飞雪，庄钊文. 卫星导航系统信号的多普勒特性分析 [J]. 国防科技大学学报，2003，25 (5)：48 - 51.
[2] 谢　钢. GPS 原理与接收机设计 [M]. 北京：电子工业出版社，2009.
[3] Tsui J B Y. Fundamentals of global positioning system receivers [M]. Wiley-Interscience, 2000: 30 - 40.
[4] Alii I, Al-Dhahir N, Hersheyj E. Doppler characterization for LEO satellites [J]. Communications, IEEE Transactions on, 1998, 46 (3): 309 - 313.
[5] Montenbruck O, Gill E. Satellite orbits: models, methods and applications [M]. Springer Science & Business Media, 2012: 36 - 39.
[6] 张爱国，刘梦鑫，陈美连，等. GPS 卫星小高度角下多普勒频移的实验分析 [J]. 厦门理工学院学报，2011，19 (1)：30 - 33.
[7] 王克锋，何　宇，赵东杰. 卫星通信多普勒频移计算方法研究 [J]. 全球定位系统，2007，31 (6)：38 - 41.
[8] 张骞丹，田红心. GPS 系统多普勒频移估算的研究 [J]. 无线电工程，2008，37 (4)：21 - 23.
[9] Amriri S, Mehdipour M. Accurate Doppler frequency shift estimation for any satellite orbit [A] . Recent Advances in Space Technologies, 2007. RAST'07. 3rd International Conference on [C]. IEEE, 2007: 602 - 607.

文章编号:1671-4598(2016)05-0284-03 DOI:10.16526/j.cnki.11—4762/tp.2016.05.078 中图分类号:TP311.5 文献标识码:A

基于隐马尔可夫链模型的软件维护性评估方法研究

郝学良，朱小冬，叶 飞
（军械工程学院，石家庄 050003）

摘要：针对当前软件可维护性评估主观性强，可操作性弱等问题，提出了定量描述维护性的维护时间统计概率描述方法，引入隐马尔可夫链（HMC）模型对维护性状态变迁过程进行描述，以可度量的维护性内部属性影响因素集量化值为观测序列，以维护时间统计概率为状态序列，构造了反映可维护性状态转移的HMC模型；收集配置管理库中软件模块历史维护时间从而确定完成维护任务频率来估计软件维护性初始状态，利用复杂网络特性计算软件维护性影响因素集的量化值，理论上即可评估出当前软件所处的维护性状态，最后运用实例对模型进行了训练与评估；结果表明，利用模型评估出的概率与实际维护任务统计出的可维护性概率基本一致，说明该方法可行且可重复，具有一定实践意义和研究前景。

关键词：软件维护性；隐马尔可夫链模型；模糊推理；信息融合；软件维护性评估

Software Maintainability Modeling and Assessment Method Research Based on Hidden Markov Chain Model

Hao Xueliang, Zhu Xiaodong, Ye Fei
(Ordnance Engineering College, Shijiazhuang 050003, China)

Abstract: In order to meet the demand of software maintainability assessment, three-state probability description method was put forward. Hidden Markov Chain model was introduced to estimate maintainability of three-state software, and state transform model was built up. Historical maintenance time of software module was collected from configuration management database to ascertain frequency of success maintenance so as to estimate software maintainability. Affecting factors set value was computed through Fuzzy inference theory. The estimation model was trained and validated through real software case, and the result shows that this method is feasible and repeatable, and can be further studied.

Keywords: software maintainability; hidden Markov chain model; fuzzy inference; information fusion; software maintainability assessment

0 引言

软件维护性是软件质量的重要属性之一，它在很大程度上直接决定了软件的后期维护成本。提高软件产品的维护性已成为软件生产方和使用方的共识。软件维护性的评估是软件维护性控制和改进的基础，已成为近年来软件维护性研究的热点和难点。

Oman维护性指数（Maintainability Index，MI）[1]通过方程式计算出来一个经验数值，在一定程度上反映了软件维护性的好坏，但是很难在实际工程中应用。Misra（2005）[3]等基于线性回归技术的软件维护性预计模型，一定程度实现了软件设计阶段的维护性预计方法，Thwin（2005）[4]等运用人工神经网络建立了软件维护性预计模型，将人工智能方法应用到了软件度量领域；Fenton[5]等将贝叶斯网络应用到软件度量与预计研究当中，提出处理不确定性问题的方法，将专家经验与知识结合起来，实现对变量间复杂关系的建模。目前关于软件维护性评估方法的研究存在定性研究多，定量研究少；主观判断多，数据验证少的特点，仍停留在软件维护性的内部属性度量阶段，缺乏外部属性和内部属性结合的综合研究。针对以上问题，本文提出了一种基于配置管理库数据的软件可维护性评估方法，将实际维护过程中的维护性外在表现信息与维护性度量工具所试题的内部度量集结合起来，实现软件维护性的整体评估。

收稿日期:2015-09-01； **修回日期**:2015-12-08。
基金项目:装备部重点预研项目“软件保障技术”(编号不公开)。
作者简介:郝学良(1984-),男,河北省邯郸市人,博士研究生,主要从事软件工程理论与应用技术方向研究。

1 软件可维护性的定义及描述方法

1.1 基于配置管理库数据的维护性评估可行性分析

软件从用户需求到开发再到发布，在需求的导引下反复维护，更新，是一个不断循环的生命周期过程。

由于软件开发配置管理库中记录了开发过程中软件开发工作量等数据，我们可利用其中的维护时间数据进行维护性度量，以可维护度及平均功能维护时间作为软件维护性评估的指标，并在实际维护活动中统计是否能按时完成任务的频率来估计维护性状态概率，理论上可以实现软件维护性的评估。再选取相应的配置管理库中的数据生成样本库，即可实现对维护性状态估计模型进行训练。利用训练好的模型从配置管理数据库中的模块进行评估，即可实现软件可维护性评估的目的。软件的开发过程本身就可以看作是一个不断进行维护的过程，在项目开发的初级阶段，往往对软件的需求认识不够清晰，使得开发项目难于一次开发成功，即使认识足够清楚，任何个人或团体都可能会出错或疏漏，因此，在软件开发过程中出现返工再开发的情况在所难免。开发的第一次经常是试验开发，其目标在于探索可行性，弄清软件需求，而第二次的开发实际上就既是开发又可看作是对第一次的维护，常见的“演化模型”和“螺旋模型”就是这种典型的开发过程。配置管理工具正是贯穿软件开发与软件维护的一个重要的桥梁。配置管理作为软件

过程管理的重要工具，不仅可以保证软件开发活动的高效，准确完成，更起着保证软件具有良好维护性的作用。

1.2 软件维护过程中的维护性内部影响因素集分析

软件维护性影响因素主要有四大类：软件基本属性（software basic attribute）、源代码设计特性（source code attribute）、文档特性（document attribute）和管理因素（management attribute），其中软件本身的设计特性从根本上决定了软件维护性的好坏。其中，前三类因素可以利用利用粗糙集原理对软件维护性影响因素进行筛选，得到软件的维护性影响因素集包括：规模、结构化因子、复杂度、耦合度、内聚度。并利用模糊推理原理，依据软件系统复杂网络特性与影响因素集之间的关系，实现维护性影响内部因素对软件可维护性影响的定量化描述[6]。

但是对于第四类因素，管理因素，如维护机构的运行机制，人员的水平等很难体现出来。本文所构建的 HMC 模型正是基于这一最难度量的维护性影响因素进行量化。

典型的软件维护过程一般是由用户需求更改发起的。首先由用户方将更改需求明确为问题报告，提交给软件维护人员，一般是开发人员。由软件维护人员对用户需求进行分析，进一步定位到具体的功能块，在配置管理中则是定位到具体的配置项，这一步即是问题报告分析时间。之后可以制定软件维护计划，以保证有效的软件维护实施和可靠的软件维护质量。在软件维护计划中，要定义软件维护目标、人员和资源分配、组织机构和保障措施等，而且要分析和制定、确认软件维护的策略、流程和和规则、实施方法和工具等。接下来进行的工作即是对源代码进行理解，对于开发人员来说，这一步所耗费的时间一般不会太多。软件完成更改后，再进行回归测试，如果满足用户需求即可发布新版本，此时维护过程完成。

软件的维护性与软件的规模密切相关，通常情况下，规模越大，维护难度越大。同时，软件的维护性在某种程度上也取决于软件的复杂度，同等规模下，软件的设计越复杂，越难以维护。

软件维护的难易大小，成本高低，主要取决于软件维护时间，而维护时间主要包括：更改请求（问题报告）分析时间、软件理解时间、故障定位时间、更改规划及更改影响分析时间、更改实施时间、回归测试时间以及其它的管理延迟时间等。

软件维护时间的长短与目标软件基本属性（成熟度、软硬件运行环境等）、源代码设计特性（结构复杂度、代码质量等）以及文档特性（完整性、一致性）有着直接的关系，另外还与人员技术水平、辅助工具等非软件系统本身因素有关，我们将这些因素统称为管理因素。

1.3 三态可维护性状态概率描述

根据 GJB 中可维护性的定义，本文将软件的维护性状态定义为三态，分别为易维护（s_0）、可维护（s_1）和不可维护（s_2）。具体在工程实际中，假设在规定组织环境内，计划完成维护任务时间或者平均分配到每个模块上的完成时间为 T，且在一定的延误时间区间内也可算作按时完成，引入容忍因子 α，容忍延误时间 $\Delta t = T * \alpha$，实际完成维护任务时间为 t，对 3 种状态作如下定义。

1）易维护状态：

对于一个软件或功能块在规定的维护条件下如果能够提前完成维护目标，那么认为该软件或功能块处于易维护状态，根据前面对各时间的定义，即 $t \in (0,T)$；软件处于易维护状态 s_0 的概率为 $P(s_0) = P(t < T)$。

2）可维护状态：

对于一个软件或功能块如果能够在一定容忍延误程度内，按时完成维护目标但是也没有提前完成，那么认为该软件或功能块处于可维护状态 s_1，根据前面对时间的定义，有 $t \in [T, T+\Delta t)$，软件处于可维护状态的概率为 $P(s_1) = P(T \leqslant t < T + \Delta t)$；

3）不可维护状态：

对于一个软件或功能块如果在规定的条件下，不能按时完成维护目标，则认为该软件处于不可维护状态，即 $t \in (t, \infty)$，软件处于不可维护状态 s_2 的概率为 $P(s_1) = P(t > T)$。

由概率和数理统计可知，概率可以用频率来估计，假如在对某软件系统进行维护，该软件系统中的某个子模块，计划完成维护任务时间为 T，容忍延误时间 $\Delta t = T * \alpha$，在时间区间 $(0,T)$ 内有 r 次完成维护任务，达到维护目标；在时间区间 $(T, T+\Delta t)$ 内有 l 次完成维护任务，达到维护目标；其余维护时间均落于区间 $(T+\Delta t, \infty)$。

根据三态可维护性状态描述可得软件处于 3 种维护状态的概率分别为：

$$P(s_0) = P(t < T) = \frac{r}{X} \tag{1}$$

$$P(s_1) = P(T \leqslant t < T + \Delta t) = \frac{l}{X} \tag{2}$$

$$P(s_2) = P(t > T) = \frac{X - r - l}{X} \tag{3}$$

2 软件维护性状态转移模型

软件维护性既是一种内部属性，主要受规模、结构化因子、复杂度、耦合度、内聚度等内部因素影响，同时也是一种外部属性，受维护组织，人员，策略等影响。因此，在软件更改维护的过程中，会牵一发而动全身，随着各影响因素的变化，其维护性也会变化，即软件的维护性状态随着软件的演化是动态变化的，对于其维护性的评估也不能静止地去考虑。因此，引入隐马尔可夫链模型来识别多种维护性影响因素集下的软件维护性状态随着软件更改而产生的转移变迁过程。

2.1 HMC 模型在维护性评估中的应用

最简单的隐马尔可夫模型一般用一个五元组来表示：

$$\lambda = (N, M, \boldsymbol{A}, \boldsymbol{B}, \pi) \tag{4}$$

其中，N 表示状态数，记 N 个隐状态为 $\theta_1, \theta_2, \cdots, \theta_N$，$t$ 时刻马尔可夫链所处的状态为 q，且 $q \in \{\theta_1, \theta_2, \cdots, \theta_N\}$；$\boldsymbol{M}$ 为每个状态对应的可观测值数目，记 $\boldsymbol{M}$ 个观测值为 $v_1, v_2, \cdots, v_M$，t 时刻观测值为 o_t 且 $o_t \in \{v_1, v_2, \cdots, v_M\}$；隐状态的初始概率分布为 $\pi = (\pi_1, \pi_2, \cdots, \pi_M)$，其中 $\pi = P(q = \theta_i)$，$1 \leqslant i \leqslant N$；$\boldsymbol{A}$ 为状态转移概率矩阵，$\boldsymbol{A} = \{a_{ij}\}_{N \times N}$，其中 $a_{ij} = P(q_{i+1} = \theta_j \mid q_i = \theta_i)$，$1 \leqslant i, j \leqslant N$；$\boldsymbol{B}$ 为观察值概率矩阵，$\boldsymbol{B} = \{b_{jk}\}_{N \times M}$，其中 $b_{jk} = P(o_i = v_k \mid q_t = \theta_j)$。

综上，HMC 模型通过状态转移矩阵 A、观察值矩阵 B 和初始状态 π 的分布来描述双重随机过程[7]。HMC 在实际应用过程中经常会用到 3 个常用的算法，如表 1 中所示。

在解决本文所提出的维护性状态评估问题中，利用配置管理库中的维护性统计数据作为模型的初始状态，用维护性度量工具度量软件的内部维护性影响因素量化值作为观测序列，采用 Baum-Welch 算法对模型参数进行训练，对训练后的模型再应用 Forward-Backward 算法计算输出概率，即可得到软件的维护性状态概率。

2.2 软件维护性状态转移模型

软件维护性是软件的一种能力，这种能力在软件设计阶段即已初步形成，在软件的开发过程中逐步确定下来，但是由于软件的维护不像硬件维修那样简单替换，而是存在不同程度的波及效应，因此软件的影响因素集及软件的可维护性状态都会随着软件维护的递进而发生改变，这个过程即是一个隐马尔可夫状态转移过程。

1）三态维护性状态转移模型：

前面已提到把软件系统的可维护性隐性状态分为 3 个，分别为：易维护状态、可维护状态 、不可维护状态，这 3 种状态的转移过程可以图 1 来表示，纵轴表示可维护性状态的分布，横轴表示软件维护基线或维护进程的更迭。

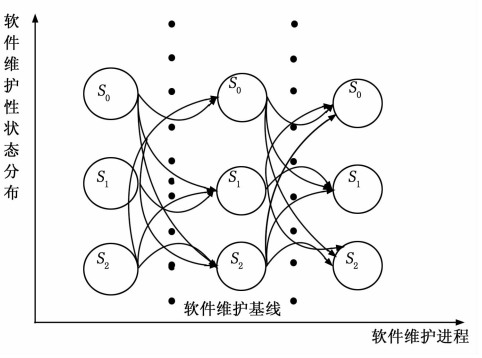

图 1 软件维护性状态转移图

结合软件三态维护性的概率表达，表 1 表达了软件维护性状态概率分布。

2）软件维护性影响因素集概率化表示：

利用粗糙集原理对软件维护性影响因素进行筛选，得到软件的维护性影响因素集包括：规模、结构化因子、复杂度、耦合度、内聚度。并利用模糊推理原理，依据软件系统复杂网络特性与影响因素集之间的关系，实现维护性影响因素对软件可维护性影响的定量化，再综合各影响因素值进行归一化处理，即得到软件维护性的观测状态值[6]。

表 1 维护性状态转移矩阵

维护次数	1	2	…	n
$P(s_0)$	π_{10}	π_{20}	…	π_{n0}
$P(s_1)$	π_{11}	π_{21}	…	π_{n1}
$P(s_2)$	π_{12}	π_{22}	…	π_{n2}

2.3 基于模糊推理的影响因素集量化

将软件规模、结构化因子、复杂度、耦合度、内聚度等 5 个影响因素与软件体系结构的复杂网络特性建立关系，并对每一个影响因素建立模糊推理系统；利用 Pajeck 分析软件模块的复杂网络特性，将复杂网络特性的取值作为模糊系统的输入，根据理论最好值与实际值的范围建立分级标准；确定输入输出变量后，以内聚度作为度量示例，建立模糊推理系统。

2.4 HMC 模型改进及训练

由于软件维护性的状态发展具有渐进的过程，建立 HMC 维护性评估模型，实质上是通过状态转移矩阵来对维护进程中的状态转移进行描述。对于这样一个矩阵，其初始状态往往不能精确获取，而是通过优化训练算法来得到状态转移矩阵，从而再通过求解 HMC 过程来实现评估。但是从理论上仍存在以下两个问题[9]：

1）HMC 初始模型的选取问题仍未完全解决。实际情况中软件的可维护性状态经常处于变化当中，初始模型的选择不当必然会对最终的评估结果造成影响。

2）HMC 模型的训练问题在整个评估模型中起着重要作用。经典的重估公式假设条件与实际应用中出入较大，可能会造成收敛速度慢甚至无法完成训练。

针对 HMC 模型存在的上述两个问题进行改进，以提高软件可维护性状态的识别速度与精度。可利用改进的参数重估公式进行多观测序列模型训练，重估公式如式（5）所示[8]。由于当前软件的维护数据较少，反映软件可维护性状态的信息也比较少，可通过历史维护数据来初步评估软件的可维护性。

现阶段的软件的维护更改过程并不是针对维护性进行的，所以软件维护性的状态是渐变的没有明确的界限。

$$\overline{\pi_i} = \sum_{l=1}^{L} \alpha_1^{(l)}(i)\beta_1^{(l)} / P(o^{(l)} \mid \lambda)$$

$$\overline{\alpha_{ij}} = \frac{\sum_{l=1}^{L}\left[\frac{1}{p_l}\sum_{t=1}^{L}\alpha_t^{(l)}(i)\alpha_{ij}b_j(o_{t+1}^{(l)})\beta_{t+1}^{(l)}(i)\right]}{\sum_{l=1}^{L}\left[\frac{1}{p_l}\sum_{t=1}^{T_l-1}\alpha_t^{(l)}\beta_t^{(l)}(i)\right]}$$

$$\overline{b_{jk}} = \frac{\sum_{l=1}^{L}\left[\frac{1}{p_l}\sum_{t=1,o_t=v_k}^{T_l}\alpha_t^{(l)}(j)\beta_t^{(l)}(j)\right]}{\sum_{l=1}^{L}\left[\frac{1}{p_l}\sum_{t=1}^{T_l}\alpha_t^{(l)}\beta_t(j)\right]}$$

$$i \in [1,n], j \in [1,n], k \in [1,m] \tag{5}$$

假设软件系统由 n 个模块组成，记为 $N=(1,2,\cdots,n)$；每个模块都具有 m 种状态，记为 $M=(1,2,\cdots,m)$；各模块的可维护性状态概率记为 P_{ij}，显然有：

$$\sum_{j=1}^{m} P_{ij} = 1,(i = 1,2,\cdots,n)$$

假设软件在初始状态下处于可维护状态，则初始状态的可维护性概率分布为 $\pi_0=(0,1,0)$，状态转移概率矩阵 $\boldsymbol{A}_0$ 和影响因素观测值矩阵 $\boldsymbol{B}_0$ 采取均匀方法选取，代入式（4）得到初始的 HMC 模型 $\lambda_0=(A_0,B_0,\pi_0)$。将归一化处理后的影响因素集的值作为观测序列输入到初始模型 $\lambda_0=(A_0,B_0,\pi_0)$，利用 Baum-Welch 算法得到软件的动态 HMC 模型 $\lambda_t=(A_t,B_t,\pi_t)$，利用前向－后向算法计算不同模块的影响因素集的量化值在给定模型 $\lambda_t=(A_t,B_t,\pi_t)$ 下的概率 $P(o\mid\lambda)$，反复训练，直到该概率值收敛，此时的 HMC 模型即可以用来评估软件可维护性状态。

令 $P_j(t)=P(q_t=j)$ 表示软件维护性在 t 时刻处于 j 状态的概率。根据切普曼－科尔莫哥洛夫微分方程，可以得到 $P'(t)=P(t)A$，求解该微分方程可以分别得到软件维护性处于 3 种状态的概率 $P(s_0)$，$P(s_1)$，$P(s_2)$。通过这 3 种状态概率可以对软件的维护性进行评估。

3 实例验证

3.1 维护性数据收集与模型求解

“美腾”是我课题组自行开发的虚拟维修训练软件，包含系统登录、信息建模以及信息管理等 12 个模块，其版本已更新至 V3.0。

在 V2.0 至 V3.0 的更新开发过程中，以“训练内容”模块为例，在配置管理工具中对每次维护都记录了检出（Check out）和检入（Check in）时间，与开发维护人员的沟通，剔除

（下转第 290 页）